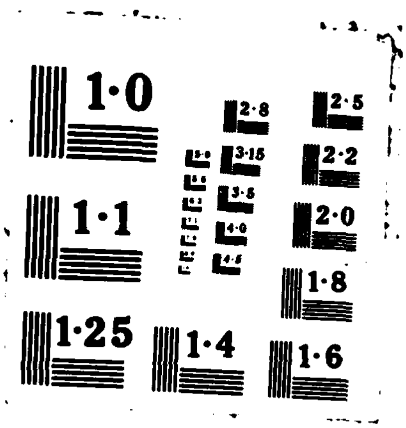
AD-A188 933 INTEGRATING ADVANCED TECHNIQUES INTO MULTIMEDIA DBMS (DATABASE MANAGEMENT SYSTEMS)(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA V Y LUM ET AL. NOV 87 NPS52-87-050 1/1

UNCLASSIFIED F/G 12/5 NL


1·0
2·8
2·5
3·15
2·2
3·5
1·1
2·0
4·0
4·5
1·8
1·25
1·4
1·6

DTIC FILE COPY

NPS52-87-050

NAVAL POSTGRADUATE SCHOOL

Monterey, California

INTEGRATING ADVANCED TECHNIQUES INTO

MULTIMEDIA DBMS

Vincent Y. Lum
C. Thomas Wu
David K. Hsiao

November 1987

Prepared for:

Chief of Naval Research
Arlington, VA 22217

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-87-050	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 52	7a NAME OF MONITORING ORGANIZATION Naval Ocean Systems Center
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) San Diego, CA
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Ocean Systems Center 8b OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N6600187WR00325
8c. ADDRESS (City, State, and ZIP Code) San Diego, CA	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: PROJECT NO: RC32510 TASK NO: H5C97-87 WORK UNIT ACCESSION NO: JO#CDB3447A01

11. TITLE (Include Security Classification)

Integrating Advanced Techniques into Multimedia DBMS

12. PERSONAL AUTHOR(S)

Vincent Y. Lum, C. Thomas Wu and David K. Hsiao

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical	FROM ______ TO ______	November 1987	32

16. SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Database management systems have been developed mainly for commercial and business processing. As such, it generally manages formatted data, rather than information, which is less structured and comes in many forms such as text, graphics, images, voices, and signals. Beyond the capability of handling multimedia information, non-business application areas call for futher requirements. Real-time processing, risk assessment, and partial answers are some of the more important requirements, especially in the military applications. In this paper, we discuss issues involved in creating such advanced database management system and propose some approaches.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Vincent Y. Lum	22b TELEPHONE (Include Area Code) 408-646-2449 22c OFFICE SYMBOL 52

Integrating Advanced Techniques into Multimedia DBMS

Vincent Lum, C. Thomas Wu, David Hsiao
Naval Postgraduate School
Department of Computer Science, Code 52
Monterey, California 93943, U. S. A.

ABSTRACT

Database management systems have been developed mainly for commercial and business processing. As such, it generally manages formatted data, rather than information, which is less structured and comes in many forms such as text, graphics, images, voices, and signals. Beyond the capability of handling multimedia information, non-business application areas call for further requirements. Real-time processing, risk assessment, and partial answers are some of the more important requirements, especially in the military applications. In this paper, we discuss issues involved in creating such advanced database management system and propose some approaches.

I. INTRODUCTION

As the DBMS[1] technology matures, its applications become more and more sophisticated and diversified. Each new application generally brings along new requirements as well. In most of the past years, DBMS research and development was concentrated on the commercial applications where the data is usually formatted. In the newer applications, such as office automation and engineering, data come in both formatted and unformatted forms, including text, graphics, and images. In some of these more advanced applications, one can expect data to be in voice and signal forms as well. To be able to handle the data in the unformatted forms increases the complexity. To be able to store the data and get them back is not a problem. To be able to do these operations efficiently is a different matter. Thus, many research projects have been started to address the new problems [BATO85, CHOC81, CHOC84, CHRI86a, CHRI86b, CROW87, DADA86, LUM82, MADI87b, MADI87c, MASU87, STON87, WOEL86]. Compared to the DBMS used in the commercial systems, these new research projects intend to incorporate into the DBMS many functions that are not generally considered necessary or even useful in commercial systems.

Data in the commercial applications generally are short, from few bytes to few hundred bytes. Unformatted data, on the other hand, generally are long, up to millions of bytes. If we were to only store and retrieve one particular unformatted data string, we would not have much of a problem, even when the data is long. We can simply store the data as one does for a file, and retrieve it as a file. Indeed, this is the normal way that one does with the graphics and image data in the engineering applications. However, to do it this way means one would not be gaining many of the benefits provided by the DBMS and their technology. The DBMS here merely

[1] The term DBMS will be used throughout the paper as both singular and plural depending on context.

behaves like a file access method. To gain the other benefits, we must provide capabilities to process unformatted data in a similar that we can do for the formatted data. Currently, we are far from this goal.

Indeed, in commercial processing, not only the data handled by the DBMS is short and formatted, but the structure of the information is also simple. It is this simplicity that allows us to use tables, as one does with the relational model, to store and process the data effectively. As we learned later in the newer applications, we found that we must have a more powerful data model to allow us to capture the necessary ingredients presented in these more complex applications. While we can still use a relational DBMS to construct the complex objects, it is generally agreed that simple enhancement to the relational systems is not sufficient. We shall discuss this point further in the section on data model.

When DBMS technology was developing, we did not forsee the broad use of this technology. We built the systems with operations that would handle the simple structures. As we cannot expect the different operations that will be needed for the different complex objects to be implemented, one cannot create a DBMS that would provide efficient operations for these unknown objects. If one just relies on the normal DBMS operations to process all the complex objects, one will not only encounter inefficiency in processing the data, but one will likely lose the advantage of the use of high level language interface provided by a DBMS.

Consider, for example, that one wants to use a DBMS to manage the data in engineering designs. Suppose one would like to display a car in a graphical form, but inside the system, the data is stored in geometric parameters, not just a simple bit string. The user would have a hard time to use the DBMS to rotate the view of the car using only the DBMS operations. What one has to do is to write programs to work with the data received from the DBMS and transform them to a suitable form. One naturally thinks that these operations should be defined and used in such a way as if the DBMS operations have been extended to have these new functions. Such a concept is referred to as abstract data type in the programming language environment [GUTT78, LISK75, SHAW80]. The same concept can be utilized in the DBMS environment as well.

Related directly to the issues just mentioned is the user interface. While much has been done in this area, much remains to be done. One reason for this is that more and more people are using DBMS to process data. The users today range from the very sophisticated to the very naive. Yet we have to accommodate all of them. Whether one should eventually have only one user interface or have many is still unclear. Nevertheless, the issue should be carefiully studied.

In fact, we really cannot expect to know what will be coming in the future in terms of applications. Neither can we see what will be the development of different technologies. We do know, however, that we want the computer to do more and more of our work. We want to have the computer to do searching and processing better than we, the humans, can. Our first step is to approximate ourselves. At least, we do this when we do not know how else to do. This means that we will have in the future more and more AI techniques incorporated into DBMS.

To achieve this, it means we must have an architecture of a DBMS to be so structured that new techniques can be 'plugged into' an existing system in some way. Such a system is currently called extensible system and is being studied in many places [BATO87a, BATO87b, LIND86, McPH87, OREN86, SHCA86, STON87]. We would like to further extend the limited goals of some of the studies to include not only some simple additional functions that are allowed to include into a system, but also sophiscated techniques like AI, signal processing, voice and sound processing, etc.

Users of any computer system are always concerned about the efficiency of the system. Thus, creators of DBMS have always tried to see how high performance can be achieved. Today's DBMS in general give good performance. In fact, in some cases, they support extremely high transaction rates. However, we have always been concerned only with commercial applications. There are other applications that have not been traditionally our concerns. For example, we know of banking applications, where the transaction volume is high, and customers are waiting in line at the terminals. We say that we want to create a real-time system so that customers would not become upset for waiting too long. Here, we mean 'real-time' to be 10, 20, or 30 seconds or more. Such timing would be quite satisfactory, as people can not react much faster.

However, in some advanced military applications, this kind of measure is totally unacceptable. For example, if the DBMS is used to store data for identifying objects, one may be forced to do split second decisions when one sees an object approaching. The same would be true if the commander of a battle group of ships wants to see how he would want to react to a battle situation. He does not always have the luxury to wait *half a minute just* to see what may come out from the computer.

Under situations like these, we need to have a DBMS that not only can handle multimedia information, but alway be able to react flexibly. In urgent cases, the response must be exceedingly rapid. At other times, one can afford to wait a while. How can we achieve such a goal? We must start from the beginning to design a DBMS that can do things this way. We do have a number of things that we must consider. First, we must see how to separate transactions that are extremely urgent from the ordinary ones. We have to store data and information that would be useful for this purpose. We most likely have to add special data structures that would be useful for this purpose.

In fact, it is definitely useful to see how one can assign risks to the transactions. That is, if we fail to response to a query, what may be the consequence. If the case is between life and death, as would be the case in some combat situations, one must deliver the answer fast, extemely fast, even if the system has not gotten the complete answer. This approach would be contrary to the current practice in DBMS applications. Today, a DBMS would either give a final answer or nothing at all. It will search its whole database to find the definite answer. It will not say, 'this is the likely answer'.

This is not the way people do things. Most of the time, people can only find approximate and incomplete answers and do the best we can with them. Even in AI studies, most researchers still try to work with the approach that will give only complete and definite answers. We think that in many applications, maybe the probabilistic or heuristic approach would be sufficient. In fact, if the data is massive and the urgency is acute, this may be the only alternative other than giving nothing at all.

To include probabilities for heuristic search in the stored information is difficult. This happens because we cannot forsee all the applications or the queries. However, in many cases, we can use the statistical approach such as sampling to assess data to derive information. To do so would require additional functions and structures in the DBMS. We would also have to develop different strategies to see how we want to handle the different queries.

One other situation that we feel important, but has traditionally been ignored, is that in many situations we cannot limit our system to be homogeneous. For example, it is not feasible for the whole government and the armed forces to possess DBMS of one kind or use only one data model in all the systems. The most likely scenario is that there will be many systems with different data models, but information is to be derived from these diverse systems and assembled in one of them for presentation. We should develop capabilities that would allow us to handle this kind of situation.

We would like to point out that the purpose of the report is to identify issues and scope out the major areas that need solutions to provide a foundation to support the applications in the various Navy programs, specifically, in the Command Systems Technology Block Program. Naturally any operational DBMS must incorporate many of the solutions already existing or becoming available in order to have a complete system. Since we have only a very small resource, we feel that it is best for us to work on those problems that are critical for us to have solutions in order to proceed further, or that are not actively pursued elsewhere in the DBMS research community. Consequently, important areas such as consistency control , recovery and reliability in a distributed DBMS environment, are not addressed as they are actively pursued in the DBMS research and development community because of their values in the commercial applications. We would also like to point out that, while we identify problems that may be specific in the Command Systems Block, it is believed that the solutions when available will have a much broader application.

In the following sections, we shall discuss some of the above issues in more details. We shall also propose some approaches in some cases.

II. DATA MODEL

The abstraction concepts our model proposed in [MADI87a] supports molecular aggregation, generalization, version, and instantiation. These concepts are necessary (but probably not sufficient in many application areas) to capture the semantics involved in the advanced application areas. We add one more abstraction concept called *medium* to the model so now it will be able to describe multimedia objects.

Molecular aggregation is the abstraction of a set of objects and their relationships into a higher-level objects [SMIT77]. This abstraction allows a viewing of objects from different levels of generality. At the executive level, for example, one may only be interested in **tank**, while at the plant level, one may be interested in the details of **tank**, that it is an aggregation of cannon, guns, caterpillar treads, etc. At the wheel engineering department level, one is now interested in the details of a wheel. The idea is to give the user only the amount of details he needs for a particular application.

The generalization concept [SMIT77] is used in our model to provide the relationship between types and their subtypes. A molecular object has a type and may consist of many subtypes. For example, a molecular object of type **surface ship** has subtypes **carrier**, **destroyer**, **frigate**, etc. In other words, a type is a generalization of a set of named subtypes. Notice that subtypes inherit their supertype's attributes and possess sets of attributes unique to them. A molecular object of type **light cruiser** inherits the attributes of **surface ship** and has a set of attributes unique to it such as number-of-depth-charges-loaded.

A version of a type (or a subtype) is defined to be a molecular object with interface details completely specified, but with implementation details in some stage of completion. A molecular object **surface ship** may have different versions such as **nuclear-powered**, **diesel-powered**, etc. These versions have the same interface, i. e. characteristics which distinguish them as surface ships. A subtype of an object, on the other hand, has different interface detail. A subtype **light cruiser** has an interface different from the subtype **carrier**. The reader may wonder how the **nuclear-powered** could be a version of **surface ship** instead of a subtype. Whether one identifies an object as a subtype or a version depends on the conceptual designer, and it is not the role of data model to determine. The reason our model includes both subtype and version concepts is because they make the modelling of application more precise. If there is only one concept -- either subtype or version -- then a subtle ambiguity and a consequent confusion may result.

An object is created by instantiation [BATO85]. Both object types and object versions can be instantiated. A version will be instantiated to provide a local working copy of a previous design, which has the implementation details specified. A type will be instantiated to provide a working copy for design work starting from scratch, that is, no implementation details are specified. In case of a battleship design, one instantiates the type **battleship** to produce a new design of a battleship from scratch. In other words, the designer has to fill in the values for number and types of armaments, displacement, etc. If the designer instantiates the version **Iowa-class**, then he need not specify the detail for the cannons, since they are already specified. Versioning allows expedient designing by reusing the old design.

We extend the described model to incorporate multimedia data. The notion is similar to version. In the extended model, we have another concept called *medium*. An object type may have several media, and a medium may be *text*, *graphic*, *image*, *voice*, *signal*, etc. So the medium specifies the mode of transmission in conveying the information from a computer to a human.

Let's say that the object **frigate** has graphic as one of its media. Then the displaying of frigate information in the graphic medium allows the user to see the visual presentation of the frigate. The regular graphics operations such as zooming, paning, rotation, scaling, etc. can be supported. Medium actually specifies the procedural detail of presenting the informational content. Now, using this graphics display of a frigate, the user can point to the cannon of the frigate and receive information about it. If the object cannon also has the graphics medium, then it too can be displayed visually (in a separate window). Otherwise, the regular information (record-based information such as size, firepower, etc.) of the cannon is presented. When there is more than one medium, then the user is prompted for the display medium.

Notice that the framework we have presented here will accomodate a new search technique in various media. We do not currently have an efficient way to recognize an object in the digitized image. For example, given a digitized image of a battle scene, it is not possible to identify objects such as tanks and armored vehicles depicted in the scene. If a new algorithm to do the task is invented, then it can be easily incorporated in our proposed framework, because the media is nothing but an abstraction of the procedural detail of displaying.

Our approach in handling multimedia objects is not to equate media to type. In the case where a media is equated to type, an object has a type graphic, text, etc., while in our case an object is of type carrier, submarine, heavy cruiser, etc., which has a different media. We believe our approach closer to the meaning of "medium", which is the mode of transferring the information. We are saying that the object has an intrinsic nature (ie. person, ship, automobile, river, etc) and this intrinsic nature can be conveyed to others via different media.

III. USER INTERFACE

Of many system requirements that the new advanced DBMS must satisfy, simplicity requirement would rank among the most critical ones. No matter how successful we are in meeting the other requirements, if the system is not simple to learn and use, no one would even bother trying to use the system. We believe a key to simplicity is a unified user interface. We would like to have a single, coherent interaction method for the manipulation of information, be it an information for marketing analysis, personnel data, or graphical display of design objects. Since the types of information manipulated in the advanced applications are complex, we feel an extension of a conventional textual language would be inappropriate. We believe the graphical interface approach has the most potential for becoming a unified interaction method that can be used for diverse application areas. Other approaches such as the natural language interface and semantic-based interface (a textual interface based on semantic data models) are simply not compatible with the way users normally function in carrying out the engineering and business activities. Many graphical interfaces are already proposed in the literature. They can be classified broadly into four groups:

a) manipulating business forms and supporting office automation [LUM82, ROWE85, SHU85, SMIT82, ZLOO82];

b) querying a database based on relational or semantic data model [BRYC86, ELMA85, GOLD85, WONG82, ZLOO77];

c) managing system resources [GOLD83, WILL84]; and

d) developing general-purpose programs [GLIN84, RAED84].

We think it is possible to create a unified graphical interface by extending/combining/modifying these proposed methods.

We proposed a graphics interface to the database and reported in [WU87, WU86]. Since we are dealing with multimedia data, the graphics mode of interaction seems to be most natural and effective. Our proposed interface provides a graphical representation for the generalization, aggregation, classification, and association concepts. We still need to add the notion of version and medium to the proposed interface to make it complete.

Our proprosed graphics user interface GLAD utilizes a bit-mapped, high-resolution graphics display terminal. The screen consists of two types of windows: *schema* and *operation window*. In the schema window, GLAD provides an elegant visual representation of real world abstraction concepts most semantic data models support: aggregation, generalization, classification, and association. In the operation windows, GLAD describes objects, displays results, and allows users to specify queries. Windows can be opened, closed, scaled, and moved at the user's will. The horizontal and vertical elevators are used to pan different portions of a display when the whole display is too large to fit in a window.

IV. SYSTEM ARCHITECTURE

As mentioned in the Introduction, because it is not possible to anticipate the advances in technology in the future, it is important to design a system architecture that would allow us to incorporate new techniques into the system as they are developed. Figure 1 is a schematic representation of the system in which we have outlined some of the key components. The Supervisor component, as the name implies, is the module that oversees and coordinates all the activities of a transaction. It invokes other components to do a specific task and assembles the information it receives from them. The Data Access component, in addition to dealing with the data from and to the storage units, actually is assumed to contain many service components to be invoked by the other processing components. For example, the locking manager, the transaction manager, the recovery unit, etc. are components in this box, although they are not explicitly illustrated in the diagram. The other components like the Formatted Data Processing unit, the Text Processing Unit, etc. are specilized components that would perform designated tasks.

In essence, compared to today's DBMS, the components like the Text Processing, the Graphics and Image Processing, etc. are generally nonexistent. Although conceptually that is all, actually there is a major difference between the proposed system and current DBMS. The difference is that current DBMS are not designed to allow its components replaced. As such, the modules' interfaces are not so clearly defined. Here, it is of paramount importance that we have very well and very clearly defined component interfaces so that each component as indicated in the diagram would be replaceable by another one in its place. This is by no means an easy task. In fact, one may question whether that is at all possible. There is substantial ground for this skepticism, particularly when we consider the complexity that may be contained in the single box called Formatted Data Processing. This one box represents a great part of today's DBMS.

While this question cannot be answered at this time, it certainly is an area where research can be and should be conducted. Further, not all the processing boxes represent the same complexity. For example, the Text Processing component is likely to be much simpler than the Graphics and Image Processing unit. In fact, we may even want to split this latter component into two: one for graphics and another one for images. The complexity of the units depend on the complexity of the structure of the information represented by the data. Images are definitely more complex than text or graphics, as it does not contain regularity as in the others. It is too early for us to say what may or may not be possible to achieve at this time. We only wish to point out that, if we do not consider this factor, we will run the risk that our system might become outdated by the time we have implemented a prototype. It will also mean that we will forever be chasing the advances in technology but never can we catch up.

Pursuing the same philosophy, it actually means that we should further divide the major components into subcomponents in such a way that would allow us to plug into the system replacement subcomponents. Indeed, an ideal is to have a system with many parts so designed that they can be replaced at will without disturbing the rest of the system.

V. INTEGRATION WITH ADVANCED TECHNIQUES

In the last section, we discussed the importance of having an architecture in the system that we can plug in new components when new techniques are developed. Let us consider here more specifically some of the new techniques that may come in the future.

The first thing that comes to mind is probably the application of artificial intelligence (AI) in DBMS. AI, by definition, is the area of research where one defines techniques that allow us to analyze and find solutions to problems as done by people. This encompasses the representation of information and knowledge, the storage and retrieval of the information, the reasoning or rationalization process, and all the other things that get involved in the process to get to an answer. As such, we can see that the area has been around as long as the existence of civilization itself. Given that this is the case, we can see that it is not possible to expect that we will have all the answers from AI researchers soon in the future.

Yet, the tie between database techniques and AI, in our view, is very close and strong. We see that when a solution to an AI approach is not known, researchers in AI will continue to pursue it until it is solved. However, once an AI technique becomes clearly defined, then the database technologists will want to incorporate that technique into a DBMS. The database researchers are system builders who constantly strive to incorporate new things into their systems to solve broader problems. After all, database researchers always have to find answers to queries posed by people. Since the power of the high level queries are practically capable of expressing most of the questions we can form, even when they are complex, database researchers in reality would want to incorporate AI techniques to their systems if these techniques are found to be useful for them to answe: queries.

If this view is correct, then the incorporation of AI techniques, or their derivatives, would be just a matter of timing. This closeness between the two areas makes it most appropriate that one should design DBMS that tightly couple techniques from these two areas together. Thus, the proposals that want to build additional layers on top of a DBMS to allow one to use a new AI technique is not a good way to build a system. This approach does not let us build the system as one piece and would cause us to lose performance. Most likely, many things will be done more than once as one layer would not know what to expect from the other.

What can be expected from the advance of AI techniques is not possible to guess. However, if we can separate the function that applies AI techniques to query processing into isolated components, we can then continue to include the new techniques as they are developed. One should note, however, when we say an AI component, we do not really mean that there is only one program or one group of programs. In fact, the AI box actually represents many, many smaller boxes. For example, we can see that, within this AI box, there may be boxes that process rules or knowledge bases. There may be a box to assess and assign risks, probabilities, or to interpolate statistics. There may be boxes to process images, or many, many other things. While we do not know exactly how many things we should have here, we would know how to deal with the additional ones if we know how to deal with some of them. Thus, we shall concentrate on only few of the well known ones in our research at this time.

Let us look at specific examples to illustrate our point. In the processing of formatted data, we know fairly well how to handle things. We have good data structures that allow us to process data efficiently. If a customer of a bank, for instance, asks for the balance of his account, the system knows how to get that answer without doing an exhaustive search, in general, because a DBMS usually has indexing methods for that purpose. In cases like this, we can actually find not just an answer, but a precise and exact answer. Let us consider what happens in the case for unformatted data.

First, let us look at the simple extension from formatted data to text data such as the one in a technical paper or in a written report. While many researchers have looked into this problem, no good solutions have been found. There are solutions, though. For example, if we want to know what documents contain information on foreign policy on Russia, we have difficulty getting a precise answer, as we have many alternatives to go about doing the query search in this case. For example, do we look for documents containing exact phrases as indicated here? Do we need a document containing words that match the phrase here? Or do we want documents dealing with the general meaning of the phrase as stated? In each of these cases, we will get a different answer. It can be much harder in one way than another. Effective and efficient retrieval is difficult because we are now dealing with information rather than just raw data. Information and knowledge is generally imprecise. Answers to queries on them are therefore generally not precise either.

When we extend from text into even more unformatted data, we have much more difficulty. Let us consider image data as an example. Assume we have stored in our system all the photographs we have taken. Now, suppose we want to find pictures that contain at least one destroyer. How are we to know which pictures contain destroyer? We first have to know what a destoyer supposed to look like. But that may not be sufficient. We need to know how to recognize a destroyer in all the different positons and angles and in groups, if we wish to be sure that we get the correct answer, namely all the pictures with at least one destoyer. This is indeed AI research. Today, AI techniques have not advanced to the state that would allow us to process queries of this kind. If such techniques were available, there is no question that they should be included in a database system to be used to process these queries in the same way that we do for the formatted data queries.

However, even in the best of the cases, we will not be able to get answers to queries as nicely as we do in the formatted data. In formatted data processing, we have precise answers. In this kind of information, it is not possible to expect the same kind of service. In real life, if we have a person looking at the pictures, there may be pictures that are not recognized by the person because of various reasons. While it may be possible for the computer to do a better job than a person, it would be a long time before the technology would advance to that state in this case.

There are, of course, other solutions. We can include a description such as keywords to each picture in the file. We can then search the key words instead. This approach, in fact, is the one adopted even in representing text documents. While we can accept solutions like this, it is definitely not the best kind. If the person who does the classification has forgotten to enter a descriptor, the system will never be able to get that particular document or picture. Since no one can think of all the descriptors pertinent to queries that are not yet defined, it is a sure thing that many pictures will not be found by the system even when they are the ones desired by the user who enters the query.

Further, such technique is not so useful for the more complex queries. For example, suppose one wishes to find the pictures that contain a carrier following a destroyer in a battle group formation. One would now have to recognize that certain motions imply something or other. For example, even though both a carrier and a destroyer are detected, if the carrier is in front, then this picture may be not the one for this query. What we want to say is this: In processing information, one needs advances in many fields, especifically in AI. Today, we use DBMS mainly to manage formatted data. Acutally, we want to develop systems that would manage information, which comes to us in many forms. However, the state of the art is such that we do not know how to handle the unformatted data well. Nevertheless, we should design DBMS in such a way that allow us to include the advances in technology in other fields.

The above argument can easily be extended to include voice prcessing or signal processing. In commercial data processing, signal processing is not very useful. In military data processing, signal processing is important. We shall not belabor the necessity of including signal processing in a DBMS, as there are numerous examples of its need in military data processing. In that environment, information is stored and derived from many forms, frequently in signals. In certain cases, the signals are transformed. For example, the noise from the engine of a submarine will likely be some graphic waveforms rather than just as sound waves. In other cases, the signal may be received in its original form. Whatever it is, the system may be required to store and retrieve it. In many cases, one may have to analyse data from many aspects in order to derive an answer to a query. By having the architecture of our system modular, we can handle this kind of processing as well.

We have earlier mentioned that risk assessment should be included in DBMS for military applications. Let us pursue this aspect a little further.

It is generally true that some queries are more pressing than others. In military situations, the inability to complete a query processing may have dire consequences. For example, we may have received a signal that something is approaching our base and we do not know exactly what it is. We try to search our database to see if anything resembling this signal. If we do not finish processing our search, numerous lives may be lost. The risk is high in this case and it should be so recognized.

One may think that we should classify queries into priorities and the system will handle them accordingly. This is a beginning. It means that a DBMS should include at least priorities in queries and shedule and process them accordingly. It is not sufficient. We should go further.

Let us suppose that we can determine the likelihood that the signal is not a threatening one. In that case, even if we do not finish processing the query, there is not much of a danger. On the other hand, if, after some analysis, the system finds that the signal is very likely to have a highly destructive effect, such result should be communicated to the user, even though the system has not completed the whole search.

Once again, it seems that AI techniques will be involved. We think that, in the AI box, there should be modules that would provide assessment on the risk factor. In some cases, the assessment may depend on the analysis of the statistics as given in the database. Whether one would need to store additional information in the system to use depends on the application and the algorithm used to generate an answer. For example, while it is possible to derive probability based on statistics, it is not possible to find any probability if the statistics and information were not there. For example, we may find an object approaching and identify the object to be hostile and threatening, but since the database does not contain the information detailing the explosive power of the object, we are not capable to assess the damage that may be caused by the object. One may not be able to correctly assign the risk factor for this situation.

In any case, risk is a valid factor to be considered and the system should include some intelligence in it so that appropriate answer can be generated.

The above approach really argues that DBMS and AI should be tightly coupled so that techniques from the two areas should be integrated. While this view may not be shared by all researchers, it seems to us most appropriate. Without DBMS technology, AI would not be able to process the voluminous data in an organized manner. Without the use of some of the AI techniques, a DBMS will not be able to process intelligently many queries. While it will always be true that AI will try to understand and find solutions to many seemingly ill-defined problems, once a solution is found, it would be appropriate to include it into a DBMS.

From the architectural point of view, as shown in Figure 1, one can say that every DBMS should have an AI component that allows the DBMS to answer queries which require logical or heuristic deductions or inductions. The DBMS, in this case will then be the superstructure that interfaces directly with the users. Such proposal is logical as DBMS traditionally deals with end-user query interfaces. On the other hand, it can also be stated that, within every AI system, there is a database system. As true to life applications always will need tens of thousands of rules and millions of facts or pieces of information, a DBMS is needed to help manage and process these rules and the information, freeing the AI component to do the logical or heuristic processing. Whether it is one way or another, it is not clear at this time. Much research is needed to answer this question. It is clear to us, however, that a strong coupling or integration is likely the direction to proceed.

VI. PERFORMANCE ENHANCEMENT

As said before, performance in military applications at times can be more demanding than commercial processing. The real-time element here can be most critical. While in commercial applications, a delay for processing a query may cause an organization a very large sum of money, in military applications, delay in response in some cases can literally mean life and death to many people. Even when a military application may not have the high transaction volume as one in commercial application, the response may be a lot more urgent. Split second timing may be necessary. Thus, it is imperative that we should see how such demands can be satisfied.

Traditionally, database management systems are designed to use moderate main memory space to hold the necessary data from secondary storage devices and use software to process the data brought in from storage. As software speed is restricted by the hardware capability, there is a hardware limit to what the designer and implementer can do to make the system faster. However, if one were to use hardware to implement some of the database functions, an increase in performance can be obtained. This is the approach of the database machine research. However, in database machines, we have not gone far enough. We have not tried to encapsulate many of the database processing algorithms in hardware form. As VLSI techniques have now been so far advanced that almost all algorithms can be captured in chips, we should take advantage of that by designing our system in such a way that we can interchange functions between hardware and software. For example, we can have a computer chip that does nothing but sorting. We can have another computer chip that does only index search. And so on and so forth. In essence, we are saying that the software and hardware boundaries are no longer distinct as it is today. With such an architecture, we will be able to take advantage of hardware advances as they come to increase performance. Naturally, by replacing software algorithms in hardware form, performance gains can be tremendous.

Another area where hardware advance can be utilized to advantage in increasing performance is the use of massive memory in a system. In recent years, main memory size has grown by leaps and bounds. It is forseeable that, in the future, one can consider it feasible to have a substantial portion of the databases residing in the main memory. It is this kind of expectation that researchers have in the past few years started research on main memory databases [DeWI84, LEHM86a, LEHM86b]. In their works, it has been shown that other than the gain in pure, raw speed over secondary storage devices, one would have to use different strategies and data structures in the

DBMS in order to get the maximum utilization of the increased memory size. In fact, pursuing further on the use of main memory to enhance performance, we will investigate the problem of allocating some primary data in the memory in expectation of the urgent demands as discussed above on risk processing. Thus, if we can identify certain group of data that is most likely needed to process the high risk queries, we can store the data in the main memory all the time. In this way, we shall be able to satisfy the urgent demands when they occur.

Just storing data in the main memory alone would not be able to satisfy the critical demand of split second responses, as frequently there is too much processing to be done and a lot of data to be processed. Having hardware embodiment of algorithms as just proposed would help. However, when the data volume to be processed is massive, one would need additional means to achieve our goal. For example, suppose one has to derive an answer statistically and the volume of data is huge. Is there something that we can do to help the system. The answer is positive. Under certain circumstances, we can build additional structures and store redundant information to facilitate the process. The paper by Srivastana and Lum [SRIV86] illustrates this point.

Statistics applications in general require the evaluation of a large amount of data to derive a value. For example, finding the mean, the deviation from the mean, the confidence level that the result is meaningful, etc. are some of the common calculations in statistical applications. Using only the raw data and not to have additional stored data for these calculations would definitely be time-consuming and requiring a lot of processing. We can, however, store additional statistical values as suggested in that paper. In this way, statistical information can be determined at a small fraction of the time needed otherwise, and queries requiring quick and urgent responses can be thus satisfied.

Other techniques that may be used to process queries requiring split second responses include the use of approximation techniques and putting out answers that are not complete. In many scientific displines such as engineering, the practice is to use approximates but on the safe side. In current database processing, we do not do anything like that. In actual human behavior, it is more often than not that we use imprecise information and satisfy ourselves with mostly approximate answers. As a first step, if not the last, maybe we should have the same approach in the system as persons actually do. Moreover, partial answers are not necessarily useless. Frequently, people can extrapolate from the partial answer to arrive at a useful decision.

VII. MULTI-DATA MODEL DBMS

In the past years, DBMS have become used in many diversified applications. It is so pervasive that almost all major systems have in it a DBMS. Moreover, as there is not any 'standard' data model, over the years, DBMS have been developed to have a number of data models and interfaces, including query languages. Not only that it is not practical, but it is really not feasible to think that, in major organizations as big as the US government, or even just one branch of the armed services, all its DBMS are homogeneous. In fact, it is expected to be exactly the opposite. That is, users cannot afford to throw the systems they have been using for years to go into a new system. This would be the case even when financially and politically realizable, because realistically it is not possible to have the manpower to make it come true. Thus, what we need is to find a way that the various DBMS can commuincate with each other. If this is not possible, we should have at least in some place a DBMS that knows how to integrate data obtained from the various systems with different models.

Figure 2 illustrates a scenario which may occur. A commander is situated in an area served directly by System A. However, to enable him to make his decision intelligently, he needs information from many systems, namely System B, C, and D. Now, as these systems contain heterogeneous DBMS, running with different hardware and software configurations and using different data models, it is necessary to find a way so that information can be obtained at the site of System A. One such way is to employ a multi-model and multi-lingual DBMS [DEMU87] at the site of System A. In this way, the user in System A may write database transactions for the other systems and obtain information from them to be converted and displayed appropriately at the user's location.

Extrapolating this scenario, one can see that it may be advantageous for the different systems, namely B, C and D, to have a copy of the muli-model and multi-lingual DBMS at their sites, coexisting with other DBMS, so that every site can receive and send information to others as required.

VIII. CONCLUSION

This paper describes an overview of a DBMS project at the Naval Postgraduate School initiated to develop an advanced DBMS that incorporates many new concepts that are not yet addressed in today's research. The concepts of risk assessment and the use of approximation or even partial answers in certain situations to answer queries are definitely new. While such approaches may not be completely appropriate in commercial applications for which current DBMS have been developed, they are most useful in other applications that occur in our normal daily life environment.

Today's DBMS manages massive amount of data in the business world very well, but they are not well suited for scientific or engineering applications and not so useful in other situations such as military applications. This occurs because the emphasis we have placed in the past is to have the DBMS handle data, but not information. People process information, but the systems we developed only process data. Information also come in different forms: fixed-size records, text, graphics, images, voices and signals. Advanced DBMS, thus, should have a capability of handling these multimedia information. Researchers in AI are more oriented toward the development of techniques to handle information (not necessarily multimedia information, though). Unfortunately, as information is very difficult to be precisely defined, advances in AI research have been slow. Since DBMS are required to be "smarter" in answering queries in advanced application areas, techniques developed in AI for processing information are directly relevant and must be incorporated into advanced DBMS whenever appropriate. It is our belief that AI techniques are one of the most important ingredient for a successful realization of an advanced DBMS.

Since it is not possible to predict something that has not been developed, and since it is not practical to build a new DBMS whenever new techniques are found, we believe that we must develop an architecture for the DBMS in such a way that it can be flexibly and broadly extended to incorporate new development in multimedia processing, inference capalibities, storage methods and others. A highly extensible architecture, with capability to handle multiple data languages and models, much beyond that has been conceived in other research programs, is indispensable for realizing an advanced DBMS.

Our report has identified many problems and a rather broad scope, even though we considered only a subset of the important areas that are pertinent to the Command Systems Technology Block Program. Our next step is to narrow down the scope further to arrive at a better focus by studying more closely the applications in this program. Meanwhile, we shall direct our efforts on object-oriented data model, user interface, and intelligent processing capabilities such as risk assessment, which are believed to be basic in the pursuit of developing an advanced DBMS to support these applications.

REFERENCES

[BATO85] Batory, D. S. and Kim, W. "Modeling Concepts for VLSI CAD Objects," *ACM Transactions on Database Systems*, Vol 10, No 3, September 1985, 322-346.

[BATO87a] Batory, D.S., J.R. Barnett, J.F. Garza, K.P. Smith, K. Tsukuda, B.C. Twichell, T.E. Wise, " Genesis: A Reconfigurable Database Management System," University of Texas at Austin Technical Report TR-86-07, March, 1986, also to appear in IEEE TOSE.

[BATO87b] Batory, D.S., "A Molecular Database Systems Technology," University of Texas at Austin Technical Report TR-87-23, June, 1987.

[BRYC86] Bryce, D. and Hull, R., "SNAP: A Grahics-Based Schema Manager," *Proceedings of Data Engineering Conference*, Los Angeles, 1986, 151-164.

[CHOC81] Chock, M., A.F. Cardenas, A. Klinger, "Manipulating Data Structures in Pictorial Information Systems", IEEE Computer, Nov. 1981, pp43-49.

[CHOC84] Chock, M., A.F. Cardenas, A. Klinger, "Database Structure and Manipulation Capabilities of a Picture Database Management System (PICDMS)", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-6, No. 4, July 1984, pp484-492.

[CHRI86a] Christodoulakis, S., F. Ho, M. Theodoridou, "The Multimedia Object Presentation Manager of MINOS: A Symmetric Approach", Proceedings of ACM SIGMOD '86, Washington, D.C., May 28-30, 1986, pp295-310.

[CHRI86b] Christodoulakis, S., M. Theodoridou, F. Ho, M. Papa, A. Pathria, "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System", ACM Trans. of Office Information Systems, Vol. 4, No. 4, Oct. 1986, pp345-383.

[CROW87] Crowder, S. and Wu, C. T., "Visual Information Management System for an Effective Performance of Office Tasks," *Proceedings of the 2nd International Conference on Human-Computer Interaction*, Honolulu, August 1987.

[DAYA86] Dayal, U., J.M. Smith, "PROBE: A Knowledge-Oriented Database Management System" in On Knowledge Base Management Systems, published by Springer-Verlag, M.L. Brodie and J.Mylopoulos, eds., 1986.

[DADA86] Dadam, P., K. Kuesport, F. Andersen, H. Blanken, R. Erbe, J. Guenaner, V. Lum, P. Pistor, G. Walch, "A DBMS Prototype to Support Extended NF^2 Relations: An Integrated View on Flat Tables and Hierarchies," *Proceedings of SIGMOD 86*, Washington, D. C., May, 1986, 356-367.

[DAYA87] Dayal, U., F. Manola, A. Buchmann, U. Chakravarthy, D. Goldhirsch, S. Heiler, J. Orenstein, A. Rosenthal, "Simplifying Complex Objects: The PROBE Approach to Modelling and Querying Them", Proceedings of German Database Conference (Datenbanksysteme in Buro, Technik und Wissenschaft), Apr. 1987, Darmsdadt, Germany.

[DEMU87] Demurjian, S. and D. K. Hsiao, "The Multilingual Database System," *Proceedings of the Third International Conference on data Engineering*, Los Angeles, Feb, 1987, 44-53.

[DeWI84] DeWitt, D., R. Katz, F. Olken, L. Shapiro, M. Stonebraker, D. Wood, "Implementation Techniques for Main Memory Database Systems", Proceeedings of ACM SIGMOD '84, Boston, Massachusetts, June, 1984, 1-8.

[ELMA85] Elmasri, R. A. and Larson, J. A., "A Graphical Query Facility for ER Databases," *Proceedings of Conference on E-R Approach*, Chicago, 1985, 236-245.

[GALL84] Gallaire, H., Minker, J. and Nicolas, J.-M. "Logic and Databases: A Deductive Approach," *ACM Computing Surveys*, Vol 16, No 2, June 1984, 153-183.

[GALL77] Gallaire, H. and Minker, J. (ed) *Logic and Data Bases*, Plenum Press, New York, 1977.

[GLIN84] Glinert, E. P. and Tanimoto, S. L., "Pict: An Interactive Graphical Programming Environment," *IEEE Computer Magazine*, Vol 17, No 11, 1984, 7-25.

[GOLD85] Goldman, K. J., Goldman, S. A., Kanellakis, P. C. and Zdonik, S. B., "ISIS: Interface for a Semantic Information System," *Proceedings of ACM SIGMOD Conference*, 1985, 328-342.

[GOLD83] Goldberg, A. and Robson, D., *Smalltalk-80, the Language and Its Implementation*, Addison-Wesley, 1983.

[GUTT78] Guttag, J. V., E. Horowitz, and D. R. Musser, "Abstract Data Types and Software Validations," *Communication of ACM*, Vol 21, No 12, 1978, 1048-1064.

[LEHM86a] Lehman, T.J., M.J. Carey, "Query Processing in Main Memory Database Management Systems", Proceedings of ACM SIGMOD '86, Washington, D.C., May 28-30, 1986, pp239-250.

[LEHM86b] Lehman, T.J., M.J. Carey, "A Study of Index Structures for Main Memory Database Management Systems", Proceedings of VLDB 1986, pp294-303.

[LIND86] Lindsay, B., J. McPherson, H. Pirahesh, "A Data Management Extension Architecture", IBM Research Report RJ5436 (55565), Dec. 19, 1986.

[LISK75] Liskov, B. H., and S. N. Zilles, "Specification Techniques for Data Abstractions," *IEEE Transactions on Software Engineering*, SE-1:1, 1975, 7-18.

[LUM85] Lum, V., P. Dadam, R. Erbe, J. Guenauer, P. Pistor, G. Walch, H. Werner, J. Woodfill, "Design of an Integrated DBMS to Support Advanced Applications," *Proceedings of International Conference on Foundations of Data Organization*, Kyoto, Japan, May, 1985, 21-31. A similar version published in *Dateubanksysteme fuer buro, Technik und Wissenschaff*, A. Blaser and P. Pistor, Editors, Springer-Verlag, 362-381.

[LUM82] Lum, V. Y., Choy, D. M. and Shu, N. C., "OPUS: An Office Procedure Automation System," *IBM System Journal*, Vol 21, No 3, 1982.

[MADI87a] Madison, D. and Wu, C. T., "An Expert System Interface and Data Requirements for the Integrated Design and Manufacturing Process," *Proceedings of the 3rd International Conference on Data Engineering*, Los Angeles, February 1987, 610-618.

[MADI87b] Madison, D. and Wu, C. T., "The *Integration* in Computer Integrated Manufacturing," *Proceedings of the International Conference on Engineering Design*, Boston, August 1987.

[MADI87c] Madison, D., Wilbur, T. and Wu, C. T., "A Data-Oriented Approach to Integrating Manufacturing Functions," submitted to *Computer In Manufacturing Engineering*.

[MANL86] Manola, F., U. Dayal, "PDM: An Object-Oriented Data Model", Proceedings of the International Workshop on Object-Oriented Database Systems, Pacific Grove, CA, Sept. 1986.

[MASU87] Masunaga, Y., "Multimedia Databases: A Formal Framework", Proceedings of IEEE Computer Society Office Automation Symposium, Gaithersburg, MD, Apr. 27-29, 1987, pp36-45.

[McPH87] McPherson, J., H. Pirahesh, "An Overview of Extensibility in Starburst", IBM Research Report RJ5599 (56909), Apr. 9, 1987.

[OREN86] Orenstein, J.A., D. Goldhirsch, F.A. Manola, "The Architecture of the PROBE Database System", Computer Corporation of America Working Paper 142.

[RAED84] Raeder, G., "Programming in Pictures," Ph.D. Dissertation, University of Southern California, November 1984.

[ROWE85] Rowe, L. R., "Fill-in-the-Form Programming," *Proceedings of VLDB 85*, Stockholm, 1985.

[SCHA86] Schwarz, P., W. Chang, J.C. Freytag, G. Lohman, J. McPherson, C. Mohan, H. Pirahesh, "Extensibility in the Starburst Database System," IBM Research Report RJ5311 (54671), Sept. 23, 1986.

[SHAW80] Shaw, M, "The Impact of Abstraction Concerns on Modern Programming Languages," *Proceedings of IEEE*, 68:9, 1980, 1119-1130.

[SHU85] Shu, N. C., "FORMAL: A Forms-Oriented, Visual-Directed Application Development System," *IEEE Computer Magazine*, Vol 18, No 8, 38-49.

[SMIT77] Smith, J. M. and Smith, D. C. P. "Database Abstractions: Aggregation and Generalization," *ACM Transactions on Database Systems*, Vol 2, No 2, June 1977, 105-133.

[SMIT82] Smith, D. C., Irby, C., Kimball, R. and Harslem, E. "The Star User Interface: An Overview," *Proceedings of National Computer Conference*, 1982, 515-527.

[SRIV86] Srivastava, J. and V. Lum, "A Tree Based Statistics Access Method (TBSAM) for Univariate Analysis," *IBM Research Report* RJ 5399 (55188), Nov, 1986.

[STON87] Stonbraker, M., L.A. Rowe, "The POSTGRES Papers", University of California at Berkeley Technical Memorandum No. UCB/ERL M86/85, June 25, 1987.

[WOEL86] Woelk, D., W. Kim, W. Luther, "An Object-Oriented Approach to Multimedia Databases", Proceedings of ACM SIGMOD '86, Washington, D.C., May 28-30, 1986, pp311-325.

[WOEL87] Woelk, D., W. Luther, W. Kim, "Multimedia Applications and Database Requirements", Proceedings of IEEE Computer Society Office Automation Symposium, Gaithersburg, MD, Apr. 27-29, 1987, pp180-189.

[WONG82] Wong, H. K. T. and Kuo, I., "GUIDE: Graphical User Interface for Database Exploration," *Proceedings of VLDB 82*, Mexico City, 1982, 22-32.

[WU86] Wu, C. T., "A New Graphical User Interface for Accessing a Database," *Proceedings of Computer Graphics Tokyo '86 Conference*, Tokyo, April 1986.

[WU87] Wu, C. T., "GLAD: Graphics LAnguage for Database," *Proceedings of the 11th International Computer Software and Applications Conference*, Tokyo, October 1987.

[ZLOO77] Zloof, M. M., "Query-by-Example: A Database Language," *IBM System Journal*, Vol 16, No 4, 1977, 324-343.

[ZLOO82] Zloof, M. M., "Office-by-Example: A Business Language that Unifies Data and Word Processing and Electronic Mail," *IBM System Journal*, Vol 21, No 3, 1982.

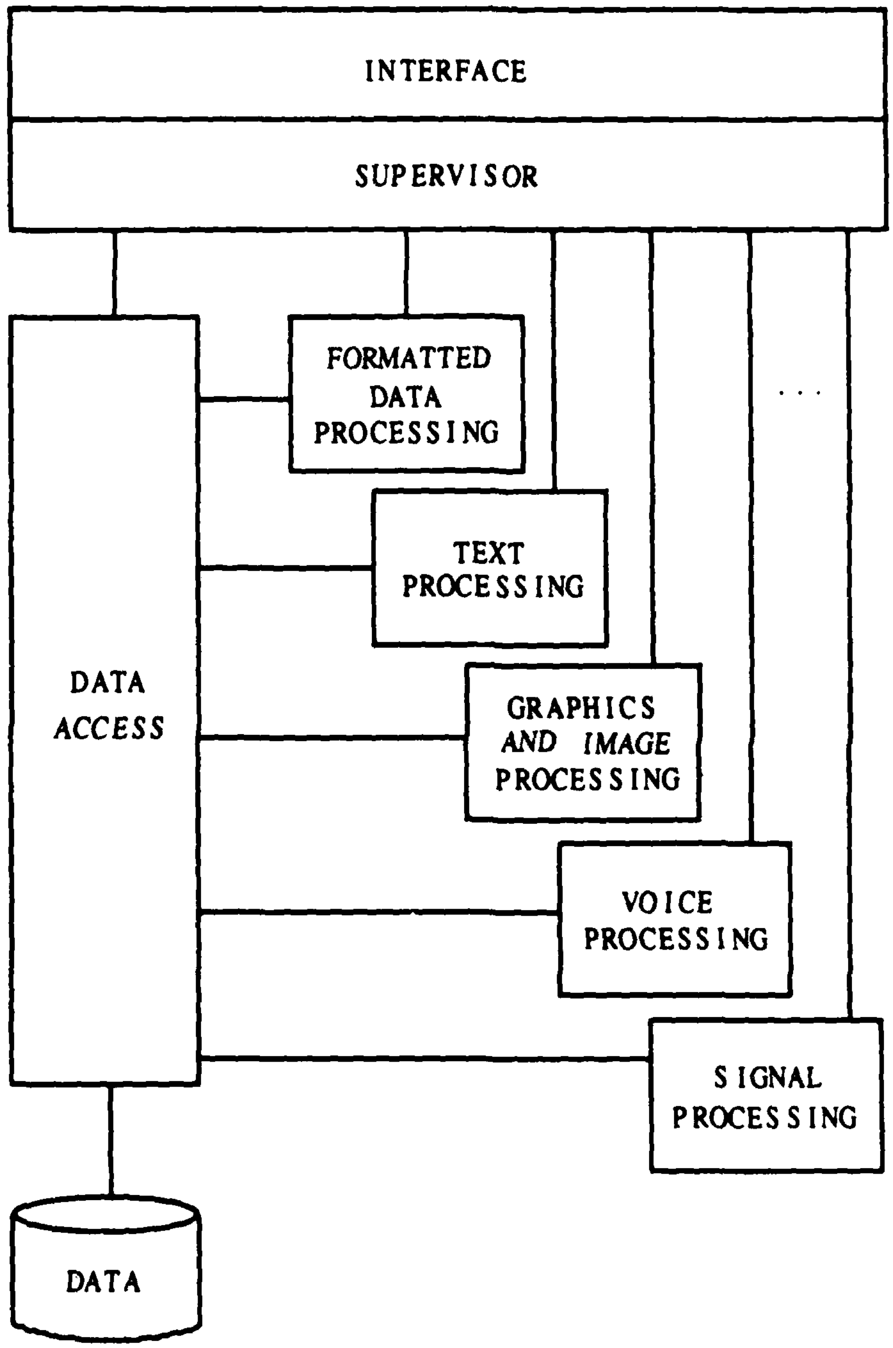


FIGURE 1 SCHEMATIC ARCHITECTURE

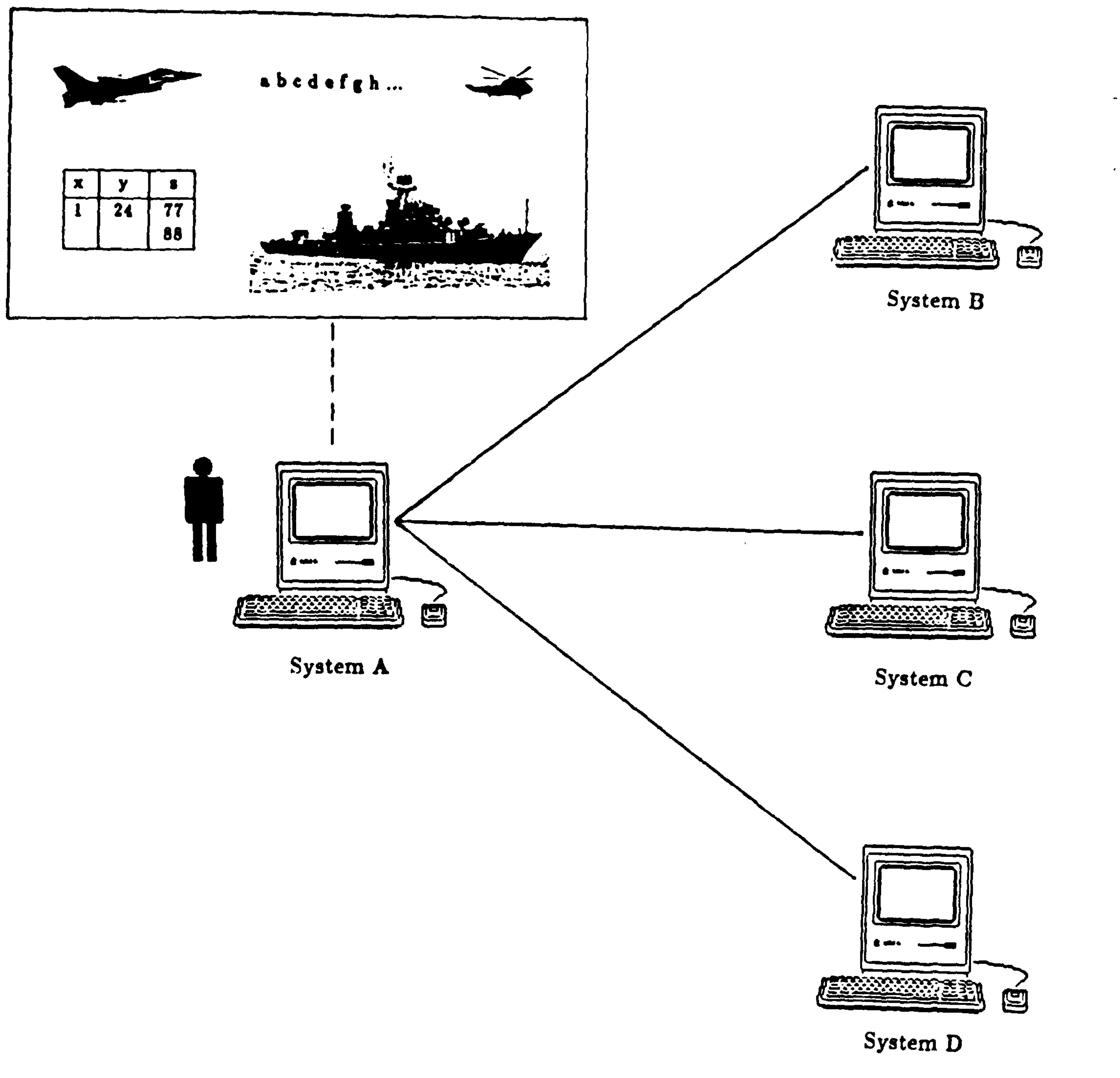


Figure 2 Distributed Environment

Distribution List

SPAWAR-3242 Attn: Phil Andrews Washington, DC 20363-5100	1
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Center for Naval Analyses 4401 Ford Ave. Alexandria, VA 22302-0268	1
Director of Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1
John Maynard Code 402	1
Command and Control Departments Naval Ocean Systems Center San Diego, CA 92152	1
Dr. Sherman Gee ONT-221 Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217-5000	1
Leah Wong Code 443 Command and Control Department Naval Ocean Systems Center San Diego, CA 92152	1

END
FILMED
MARCH, 1988
DTIC